Listing of Claims:

1. (Currently amended) A method for monitoring updates to a software repository in a multi-author software design environment, comprising:

downloading ~~a baseline~~ snapshots of an ~~baseline version of a software~~ interface to a stored software component, the snapshots having respective versions, said snapshots ~~baseline version~~ obtained over a network from a software repository via an application programming interface;

~~storing the baseline snapshot;~~

~~assembling an updated snapshot of an updated version of the software interface, said updated version obtained over the network from the software repository via the application programming interface;~~

comparing successively downloaded versions of the ~~updated~~ snapshots ~~to the baseline snapshot~~ to detect at least one difference between them ~~updated version and the baseline version~~;

rating each detected difference according to a backward compatibility metric;

determining an overall backward compatibility score of the updated version based on the difference ratings; and

issuing an alert message to registered authors of the software design environment. ~~containing the overall backward compatibility.~~

2. (Currently amended) The method of claim 1, wherein the alert message is issued only when the overall backward compatibility score indicates the updated version is not backward compatible.

3. (Previously presented) The method of claim 1, wherein the compatibility metric comprises a table of software modifications including backward-compatible software modifications and backward-incompatible software modifications.

4. (Previously presented) The method of claim 3, wherein the backward-incompatible software modifications include:

deleting a parameter from a subroutine; and

deleting a field from a public data structure.

5. (Previously presented) The method of claim 3, wherein the backward-incompatible software modifications include:

adding a mandatory parameter to a subroutine; and

adding a mandatory field to a public data structure.

6. (Previously presented) The method of claim 3, wherein the backward-incompatible software modifications include:

redefining an optional parameter as a mandatory parameter;

changing a parameter data type; and

changing a public field data type.

7. (Currently Amended) A method for monitoring software updates in a multi-author software design environment, comprising:

downloading snapshots of a software object where each snapshot is of a respective version;

comparing successively downloaded versions of the snapshots;

detecting at least one difference between snapshots; ~~a baseline version of a software object and an updated version of the software object;~~

rating each detected difference according to a backward compatibility metric;

determining an overall backward compatibility score of successive ~~the updated~~ versions based on the detected difference ratings;

issuing an alert message including the overall backward compatibility score to registered authors of the software design environment.

8. (Previously presented) The method of claim 7, wherein the alert message includes a summary of each detected difference.

9. (Currently amended) The method of claim 7, wherein the detecting comprises discovering that a parameter in ~~the baseline~~ a version is missing from ~~the updated~~ a successive version.

10. (Currently amended) The method of claim 7, wherein the detecting comprises discovering that a parameter is optional ~~in the baseline~~ a version, but the parameter is mandatory ~~in the updated~~ a successive version.

11. (Currently amended) The method of claim 7, wherein the detecting comprises discovering that a parameter ~~in the baseline~~ a version is defined as a different data type ~~in the updated~~ a successive version.

12. (Currently Amended) A method for monitoring updates to a software object repository in a multi-author software design environment, comprising:

downloading snapshots of a software object where each snapshot is of a respective version;

comparing successively downloaded versions of the snapshots;

detecting at least one difference between ~~a baseline~~ version of an object interface and ~~an updated~~ successive version of the object interface; and

issuing an alert to registered authors of the software design environment when at least one of the detected differences indicates the ~~updated~~ successive version is not backward compatible.

13. (Previously presented) The method of claim 12, wherein the object interface comprises a list of components published by a software object residing in the object repository, said components including object properties and object methods.

14. (Currently Amended) A method for ~~comparing~~ monitoring updates to software interfaces in a multi-author software design environment, comprising:

taking a ~~first~~ snapshot of a first software interface of a stored software component;

detecting at least one difference between the ~~first~~ snapshot and a ~~second~~ snapshot of a second software interface of a second stored software component; ~~and~~

rating each detected difference according to a backward compatibility metric; and

issuing an alert to registered authors of the software design environment when at least one of the detected differences indicates the first software interface is not backward compatible with respect to the second software interface.

15. (Previously presented) The method of claim 14, further comprising:

rating each detected difference according to a predetermined difference metric;

determining an overall difference between the first software interface and the second software interface based on the difference ratings; and

incorporating the overall difference into the alert message.

16. (Previously presented) The method of claim 14, further comprising:

assigning a user to the first snapshot; and

issuing the alert message to the user.

17. (Previously presented) The method of claim 14, further comprising:

incorporating a summary of each detected difference into the alert message.

18. (Previously presented) The method of claim 14, wherein a snapshot comprises a table having at least one record, said record including at least one attribute.

19. (Previously presented) The method of claim 18, wherein said record describes an object property.

20. (Previously presented) The method of claim 19, wherein said attribute indicates a data type of the object property.

21. (Previously presented) The method of claim 19, wherein said attribute indicates a data length of the object property.

22. (Previously presented) The method of claim 18, wherein said record describes an object method.

23. (Previously presented) The method of claim 18, wherein said record describes an object method parameter.

24. (Previously presented) The method of claim 23, wherein said attribute indicates a data type of the object method parameter.

25. (Currently amended) A computer programmed to monitor versions of a software interface in a multi-author software design environment, comprising:

means to download snapshots of a software interface where each snapshot is of a respective version;

means to compare successively downloaded versions of the snapshots;

~~means to assemble a baseline snapshot of a baseline version of a software interface;~~

~~means to store the baseline snapshot;~~

~~means to assemble an updated snapshot of an updated version of the software interface;~~

means to ~~compare the updated snapshot to the stored baseline snapshot to~~ detect at least one difference between the snapshots; ~~updated version and the baseline version;~~

means to determine an overall backward compatibility of a successive snapshot ~~the updated version~~ based on the detected differences; and

means to issue an alert message containing the overall backward compatibility to registered authors of the software design environment.

26. (Previously presented) The computer of claim 25, further comprising:

means to rate each detected difference according to a backward compatibility metric.

27. (Previously presented) The computer of claim 26, further comprising:

means to incorporate the ratings and the detected differences into the alert message.

28. (Currently Amended) A ~~machine~~-computer-readable medium having stored thereon a plurality of instructions for monitoring a software interface in a multi-author software design environment, the plurality of instructions comprising instructions to:

download a snapshot of a first software interface;

~~take a first snapshot of a first software interface;~~

detect at least one difference between the first snapshot and a second snapshot of a second software interface;

rate each detected difference according to a backward compatibility metric;

determine an overall backward compatibility of the first software interface with respect to the second software interface, based on the difference ratings; and

issue an alert message including the overall backward compatibility to registered authors of the software design environment.

29. (Previously presented) A computing apparatus programmed to execute the plurality of instructions stored on the machine-readable medium of claim 28 in response to an external trigger.

30. (Previously presented) The computing apparatus of claim 29 wherein the external trigger comprises a received notification of an update to the second software interface.

31. (Previously presented) The computing apparatus of claim 29 wherein the external trigger comprises a received notification of a scheduled event.

32. (Currently amended) A computer system, including:

a processor coupled to a network;

an electronic file storage device coupled to the processor; and

a main memory coupled to the processor, the main memory containing a plurality of executable instructions to implement a method for monitoring software interfaces, the method comprising:

accessing a first software interface stored on a network server;

taking a first snapshot of the first software interface;

storing the first snapshot on the electronic file storage device;

accessing a second software interface stored on the network server;

taking a second snapshot of a second software interface;

detecting at least one difference between the first snapshot and the second snapshot;

rating each detected difference according to a predetermined difference metric;

determining an overall difference between the first software interface and the second software interface based on the difference ratings; and

issuing an alert to registered authors of the software design environment when the overall difference indicates the first software interface is not backward compatible with respect to the second software interface.

33. (New) The method of claim 1, wherein the snapshot comprises:

the relevant data objects, subroutines, and associated subroutine attributes;

where the proceeding are compiled for a selection of interfaces as well as any interfaces that are associated with the selected interfaces.

34. (New) The method of claim 1, further providing an option to store the snapshots in a snapshot history.

35. (New) The method of claim 1, wherein the alert message is issued if no difference is detected between the two snapshots being compared.